(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 March 2001 (01.03.2001)

PCT

(10) International Publication Number
WO 01/14960 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/US00/22904

(22) International Filing Date: 18 August 2000 (18.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/379,556 23 August 1999 (23.08.1999) US

(71) Applicant *(for AT, BE, CH, CN, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, JP, KR, LU, NL, PT, SE only)*: **KONINKLIJKE PHILIPS ELECTRONICS N.V.** [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(71) Applicant *(for MC only)*: **PHILIPS SEMICONDUCTORS, INC.** [US/US]; Legal Department, 811 E. Arques Avenue, MS 54, Sunnyvale, CA 94088-3409 (US).

(72) Inventor: **DALMASSO, Jean-Pierre**; "Le flaubert" Bat. B, 1066, chemin Rabiac Estagnol, F-06600 Antibes (FR).

(74) Agent: **GRZELAK, Keith, D.**; Wells, St. John, Roberts, Gregory & Matkin, P.S., Suite 1300, 601 West First Avenue, Spokane, WA 99201-3828 (US).

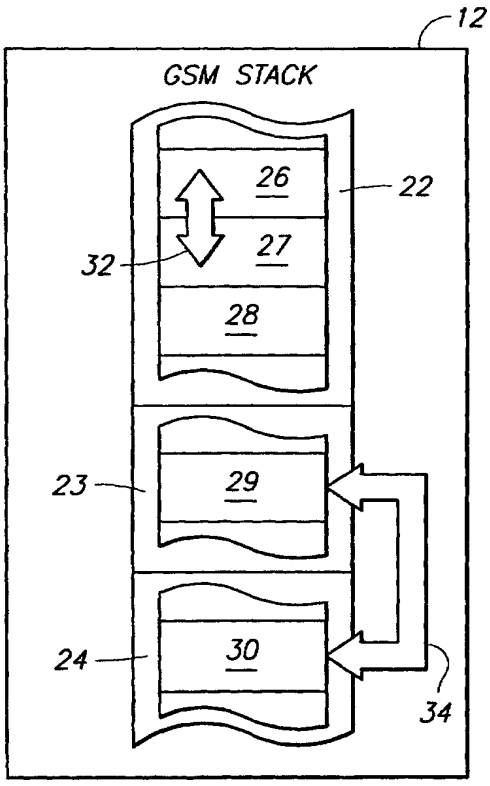
(81) Designated States *(national)*: CN, JP, KR.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: GENERIC INTERFACE FOR A SOFTWARE MODULE



(57) Abstract: An apparatus for controlling a software module is provided which includes a software module having at least one input port and an output port, the module operative to perform a task. A linking interface communicates with the module and is operative to control a state of the software module. Memory is operative to store a plurality of inputs that configure states for the module, and processing circuitry communicates with the memory and the interface and is operative to control the state of the module by delivering one of the inputs from the memory to the linking interface. A method is also provided.

WO 01/14960 A2

DESCRIPTION

GENERIC INTERFACE FOR A SOFTWARE MODULE

Cross-Reference to Related Application

This application claims the benefit of U.S. Patent Application Serial No. 09/379,556 filed 23 August 1999.

Technical Field

This invention relates to software and the control of software modules. More particularly, this invention relates to an apparatus and method for controlling a software module with a generic interface, such as performing a synchronous task like the control of a hardware chipset within a device, for example, within a portable telecommunications device.

Background Art

The utilization of software has grown significantly in recent years. The advent of personal computers (PCs) has led to a proliferation in the number of software packages and solutions. Additionally, many portable electronic devices and telecommunications devices include integrated circuit chipsets that contain processing circuitry and memory which take advantage of software solutions to deliver functionality to a user. For example, one digital mobile telephone system referred to as Groupe Special Mobile (GSM) utilizes both software modules and hardware modules to deliver functionality over a digital mobile telephone system.

GSM handsets use complex signaling protocols which are required for proper operation of a GSM handset. The complex signaling protocols are implemented in a GSM Protocol Stack which provides a complete multiband handset solution when coupled with one or more layers of internal operating software. The layered operating software is a layered structure that closely follows a software architecture that is defined by a specification for the GSM standard. Specific architecture and developments for the GSM standard are made by the European Telecommunications Standard Institute (ETSI) via a Special Mobile Groupe (SMG) technical committee. A set of technical specifications is published by the Publications Office of ETSI, and such publication is herein incorporated by reference as though set forth in full.

The GSM standard has many applications including utilization on hand-held devices, portable devices, transportable devices, portable phone systems, computer systems, modems, and other systems where the utilization of GSM mobile station functionality is desired. In order to add GSM architecture functionality to many devices,

software modules and hardware modules are utilized. However, the utilization of software modules presents problems during the design of such devices.

Disclosure of the Invention

An apparatus and method are provided for controlling a software module. The control is achieved by sending events to a software module. The events are used to control specific behavior of the software module. The software module performs a synchronous task; e.g., the control of a hardware chipset for a GSM handset.

According to one aspect of the invention, an apparatus for controlling a software module is provided which includes a software module having at least one input port and an output port, the module operative to perform a task. A linking interface communicates with the module and is operative to control a state of the software module. Memory is operative to store a plurality of inputs that configure states for the module, and processing circuitry communicates with the memory and the interface and is operative to control the state of the module by delivering one of the inputs from the memory to the linking interface.

According to another aspect of the invention, a software module control system for controlling a software module having at least one input port and an output port is provided. An interface is configured to communicate with a software module and is operative to control an operating state of the software module. Processing circuitry communicates with the interface and is operative to selectively control the operating state of the software module by delivering an input that configures an operating state of the software module.

According to yet another aspect, a method for controlling a software module is provided which includes a software module having at least one input port and an output port, wherein the software module is operative to perform a task. The method also includes a plurality of events that configure the software module into respective operating states; applying one of the events to the software module to configure the operating state of the software module.

One advantage of the present invention is improved control of a software module.

Brief Description of the Drawings

Preferred embodiments of the invention are described below with reference to the following accompanying drawings.

Fig. 1 is a block diagram of a GSM handset and mobile station system utilizing an apparatus for controlling an internal software module according to Applicant's invention.

Fig. 2 is a schematic block diagram illustrating the manner in which software modules are organized within one or more layers of a computer program, in this case how a GSM stack is organized within a mobile phone handset.

Fig. 3 is a schematic block diagram illustrating the manner in which events are sent to a synchronous software module in order to control the software module.

Fig. 4 is a flowchart illustrating operation of the method and system according to one aspect of Applicant's invention.

Best Modes for Carrying Out the Invention and Disclosure of Invention

For purposes of this disclosure, "GSM" is understood to refer to GSM900 and derivative standards DCS 1800 and PCS 1900. GSM is one exemplary environment where software modules can be controlled using the apparatus and method of Applicant's invention. However, it is understood that the features of Applicant's invention can be used to control software modules within any environment that requires the control of a software module.

Also for purposes of this disclosure, a "software module" is understood to refer to a logically self-contained and discrete part of a larger program, and can include objects, functions, procedures, routines and applications. A complete application program is formed from one or more software modules, or program modules. A software module is configured to accept logical inputs and logical outputs. Internally, the software module carries out a set of processing actions which are reflected in the output. Typically, a software module will be designed with only one input and one output which form a single entry point and exit point. However, for purposes of implementing Applicant's invention it is understood that a software module can have more than one input and output. Typically, a software module is used to implement a subroutine where there exists only a single input and output, or entry and exit points. Software modules are a product of modular programming which enables breakdown of complex tasks into smaller and simpler subtasks which facilitates the design and test of complex functional programs. However, the interactions between individual software modules has heretofore been very restricted. Although this greatly simplifies the understanding as to how a program works, improved techniques for communicating and controlling software modules pursuant to Applicant's invention are greatly needed.

The control of software modules is a desirable feature for many software applications, including the control of a software module located within a GSM stack of a GSM handset or mobile phone. Figure 1 is a block diagram of a GSM handset and mobile station system that includes a GSM stack with software layers and software

modules that benefit by utilizing Applicant's apparatus and method for controlling particular software modules. In order to carry out such control, an interface is provided that sends specific events to software modules in order to implement control of such modules.

A GSM handset 10 includes a GSM stack 12 and internal hardware components carried on one or more circuit boards 14 and 16. Processing circuitry 18 is included on boards 14 and 16 along with memory 20. In order to implement GSM architecture digital communications, it is understood that circuit boards 14 and 16 also include additional hardware such as a digital signal processor, transceiver circuitry, a source coding bank and a data codec. However, such GSM hardware or structure is not necessary to understand the implementation of Applicant's invention, and therefore further details will not be described in order that the features of Applicant's invention are not obscured in the process.

As shown in Figure 1, GSM handset 10 is a portable electronic communications device that utilizes a software package that includes layers and modules that will benefit from implementation of Applicant's invention. More particularly, handset 10 includes software in the form of GSM stack 12. Stack 12 includes a plurality of software layers identified for purposes of illustration as layers 22-24. Such software layers are used to subdivide a software stack into discrete components so that design and test can be subdivided into various different organizations or groups.

Generally, functions in higher, or outer, layers call functions in lower, or inner, layers. For example, layer 1, identified by reference numeral 22, might be a lower level layer that is developed by Company A, while mid-level layer 23 is developed by Company B. In this manner, the complexity of a software package or solution can be divided so that design tasks can be split up and handed out to specific organizations/individuals based upon expertise and availability to contribute to the design process. Typically, higher layers are implemented in terms of the functionality of lower layers. By breaking an application program into layers, clean interfaces are provided which facilitates design and testing.

Furthermore, each layer 22-24 of the GSM software stack 12 is subdivided into several software modules. A software module is a self-contained software component that cooperates interactively with a larger program or system. Typically, a software module is designed to implement a specific task within a larger software program. Accordingly, software modules are utilized in modular programming where the design of a program is broken down into individual components, or modules, that are each

capable of being independently programmed and tested. Modular programming has become a necessity when designing, developing, and maintaining relatively large software projects. A similar approach has been undertaken with respect to the use of self-contained hardware modules. The field of object-oriented programming (OOPS) has evolved from modular programming, and a set of formal rules has been laid out for developing self-contained software modules.

A schematic block diagram illustrating the manner in which software modules are organized within one or more layers of a computer program is shown and described below with reference to Figure 2.

As shown in Figure 2, GSM stack 12 is broken into several layers 22-24, and each layer may be broken down into a plurality of individual software modules. For example, layer 22 is shown broken down into individual software modules 26-28. Layer 23 includes module 29, whereas layer 24 includes module 30. A link is provided between software modules 26-27 via an interface 32. A similar link is provided by interface 34 between modules 29 and 30. It is further understood that additional interfaces (not shown) can be provided between any of modules 26-30. Interface 32 comprises a linking interface that communicates with associated software modules 26 and 27. Interface 32, in operation, is operative to control the behavior of software modules 26 and/or 28.

As shown in Figure 2, interface 32 is provided between two software modules 26 and 27 that are located within the same software layer 22 (here identified as "Layer 1"). However, it is understood that interface 32 can also be provided between software modules present in different software layers; e.g., software modules present in "Layer 1" and "Layer 2" identified by reference numerals 22 and 24. Layers 22 and 24 are both contained within the GSM stack 12.

Interface 34 illustrates a link between module 29, located in a lower layer of GSM stack 12, and module 30, located in an upper layer of GSM stack 12. Such link can be implemented over a logical input of a software module. In one case, the link is implemented over a single input of a software module. In another case, the software module has a plurality of inputs, and a dedicated input is provided for the link, or interface.

By providing software modules with interfaces, a software developer can develop a lower level module, such as module 29, for a program stack 12, while upper levels, or layers, in stack 12 are being developed by other companies. At the same time, the control features of Applicant's invention, enabled via interface 34, provide for control

of specific software modules via a series of "EVENTS". These specific "EVENTS" can be used to put a software module into a specific operating state.

More particularly, a software module can be controlled by events which enable the placement of a software module into a specific state. For example, each software module can be a specific task that is to be performed within the software stack. Two events can be defined for controlling the state of the software module. For example, one event can drive the software module to an active state, while another event can drive the software module to a deactivated state. Hence, events can be sent to a module which is very useful for software applications such as GSM applications since an event can be used to send a command to a software module where one event activates the state and the other event deactivates the state.

Such events are stored in memory, such as a look-up table or read-only memory (ROM). Processing circuitry, in the form of a processor, microprocessor or micro-controller, implement the control of individual software modules by delivering such events and logical inputs to a software module via a linking interface, and by receiving logical outputs from the software module. Accordingly, the processing circuitry regulates communication between software modules, and sets the control of individual software modules.

According to another example, one event can set a software module into an idle mode, while another event can set a software module into a dedicate mode. Such events can be applied to all modules in a software system, such as a GSM stack. Accordingly, the events are used to control the behavior of the respective software module.

A list of possible events is provided as follows:

A. ACTIVATE - This event is used to "awaken" a software module. After occurrence of this event, the software module can treat other events.

B. DEACTIVATE - This event is used to "deactivate" the module. After occurrence of this event, the software module does not treat other events, except for an ACTIVATE event.

C. TICK - This event is used to generate a clock for the software module. The START event and STOP event are treated after the occurrence of the TICK event.

D. START - This event is used to start the process of the software module.

E. STOP - This event is used to stop the process of the software module.

As shown in Figure 3, interface 32 enables the application of events 36 to specific software modules such as software module 26 from external sources, such as from other software modules. Each event 36 is operative to generate a change in state

for the software module. The events 36 on the list described above are identified as follows: activate event 38; deactivate event 39; tick event 40; start event 41; and stop event 42. By applying these events 38-42 selectively to a software module, the operating states of such software modules can be controlled.

For example, activate event 38 and deactivate event 39 cause "wake-up" and "sleep" states to be realized by a software module. Tick event 39 provides a clock pulse for the software module. In operation, it is required that tick event 40 precede use of start event 41 and stop event 42. Start event 41 is used to start the process of the software module, whereas stop event 42 is used to stop the process of the software module.

With respect to tick event 40, a "tick" is a unit of time, or time frame, utilized within GSM. A single "tick" has a defined unit length somewhere in the range of 6/1300 milliseconds and provides the reference clock of GSM. Everything in GSM is regulated by this reference clock, and a tick event is sent to each frame of the time frame. In this manner, the behavior of the module can be controlled synchronously.

An interface design is used to realize Applicant's invention where a software system is divided into modules. The ability is provided to link between modules, and between layers or stacks. Accordingly, the ability is provided via events to control the operating states of a software module.

Figure 4 illustrates a process flow diagram showing logic processing for configuring the operating state of a software module. Such process flow diagram is implemented on processing circuitry 18 (of Fig. 1). More particularly, Figure 4 illustrates the process of configuring the operating state of a software module. The logic flow diagram illustrates the steps implemented by processing circuitry when configuring such software module.

In Step "S1", a software module is provided having at least one input port and an output port, wherein the software module is operative to perform a task. After performing Step "S1", the process proceeds to Step "S2".

In Step "S2", one or more events are provided that configure the software module into one or more operating states. For example, a plurality of events are provided that configure the software module into respective operating states. After performing Step "S2", the process proceeds to Step "S3".

In Step "S3", the processing circuitry is used to apply one of the events to the software module in order to configure the operating state of the software module. More particularly, the processing circuitry applies one of the events to the software module

in order to configure the operating state of the software module. After performing Step "S3", the software module configuration technique of Applicant's invention terminates.

CLAIMS

1. An apparatus for controlling a software module, comprising:

a software module having at least one input port and an output port, the module operative to perform a task;

a linking interface communicating with the module and operative to control a state of the software module;

memory operative to store a plurality of inputs that configure states for the module; and

processing circuitry communicating with the memory and the interface and operative to control the state of the module by delivering one of the inputs from the memory to the linking interface.

2. The apparatus of claim 1 wherein the linking interface is operative to impart a plurality of states to the software module by retrieving a corresponding input from the memory.

3. The apparatus of claim 1 wherein one of the inputs retrieved from the memory imparts a tick state on the software module.

4. The apparatus of claim 1 wherein one of the inputs retrieved from the memory imparts a start process state on the software module, and another input imparts a stop process state on the software module.

5. The apparatus of claim 1 wherein the processing circuitry comprises a microprocessor, and the linking interface is implemented on the microprocessor.

6. The apparatus of claim 1 wherein the memory has a table of events containing one or more event entries that control the operating state of the software module.

7. The apparatus of claim 6 wherein the plurality of inputs within the memory comprise the event entries.

8. The apparatus of claim 1 wherein the linking interface comprises a communication linkage coupled between the processing circuitry and the software module.

9. The apparatus of claim 1 wherein the linking interface is operative to communicate between a pair of software modules.

10. The apparatus of claim 1 wherein the software module comprises a plurality of software modules, the linking interface communicating with each of the software modules.

11. The apparatus of claim 1 wherein the software module is contained within a layer of a software program.

12. The apparatus of claim 1 wherein a first software module is contained within a first layer of a software program, and a second software layer is contained within a second layer of the software program.

13. A software module control system for controlling a software module having at least one input port and an output port, comprising:

an interface configured to communicate with a software module and operative to control an operating state of the software module; and

processing circuitry communicating with the interface and operative to selectively control the operating state of the software module by delivering an input that configures an operating state of the software module.

14. The control system of claim 13 wherein the input comprises an event that is delivered to the software module through the interface in response to the processing circuitry.

15. The control system of claim 14 wherein the event comprises an activate state used to awaken the software module.

16. The control system of claim 14 wherein the event comprises a deactivate state used to deactivate the module.

17. The control system of claim 14 wherein the event comprises a tick state used to generate a clock signal for the software module.

18. The control system of claim 14 wherein the event comprises a start state operative to initiate operation of a task by the software module.

19. The control system of claim 14 wherein the event comprises a stop state configured to stop processing of a task by the software module.

20. The control system of claim 13 wherein the interface comprises a linking interface coupled between the processing circuitry and a software module.

21. The control system of claim 13 wherein the interface communicates between a pair of software modules.

22. The control system of claim 21 wherein a first software module is resident within a first software layer of a software program, and the second software module is resident within a second layer of the software program.

23. The control system of claim 21 wherein each software module is resident within a layer of a software program.

24. The control system of claim 13 further comprising a software module having at least one input and an output port, the module operative to perform a synchronous task.

25. The control system of claim 24 wherein the synchronous task comprises controlling operation of a hardware chipset.

26. A method for controlling a software module, comprising:

providing a software module having at least one input port and an output port, wherein the software module is operative to perform a task;

providing a plurality of events that configure the software module into respective operating states; and

applying one of the events to the software module to configure the operating state of the software module.

27. The method of claim 26 wherein the software module is configured into an activate state that awakens the software module.

28. The method of claim 26 wherein the software module is configured into a deactivate state that deactivates operation of the module.

29. The method of claim 26 wherein the software module is configured into a tick state comprising an operating clock for the software module.

30. The method of claim 26 wherein the software module is configured into a start state that starts a process of the software module.

31. The method of claim 26 wherein the software module is configured into a stop state that stops the process of the software module.

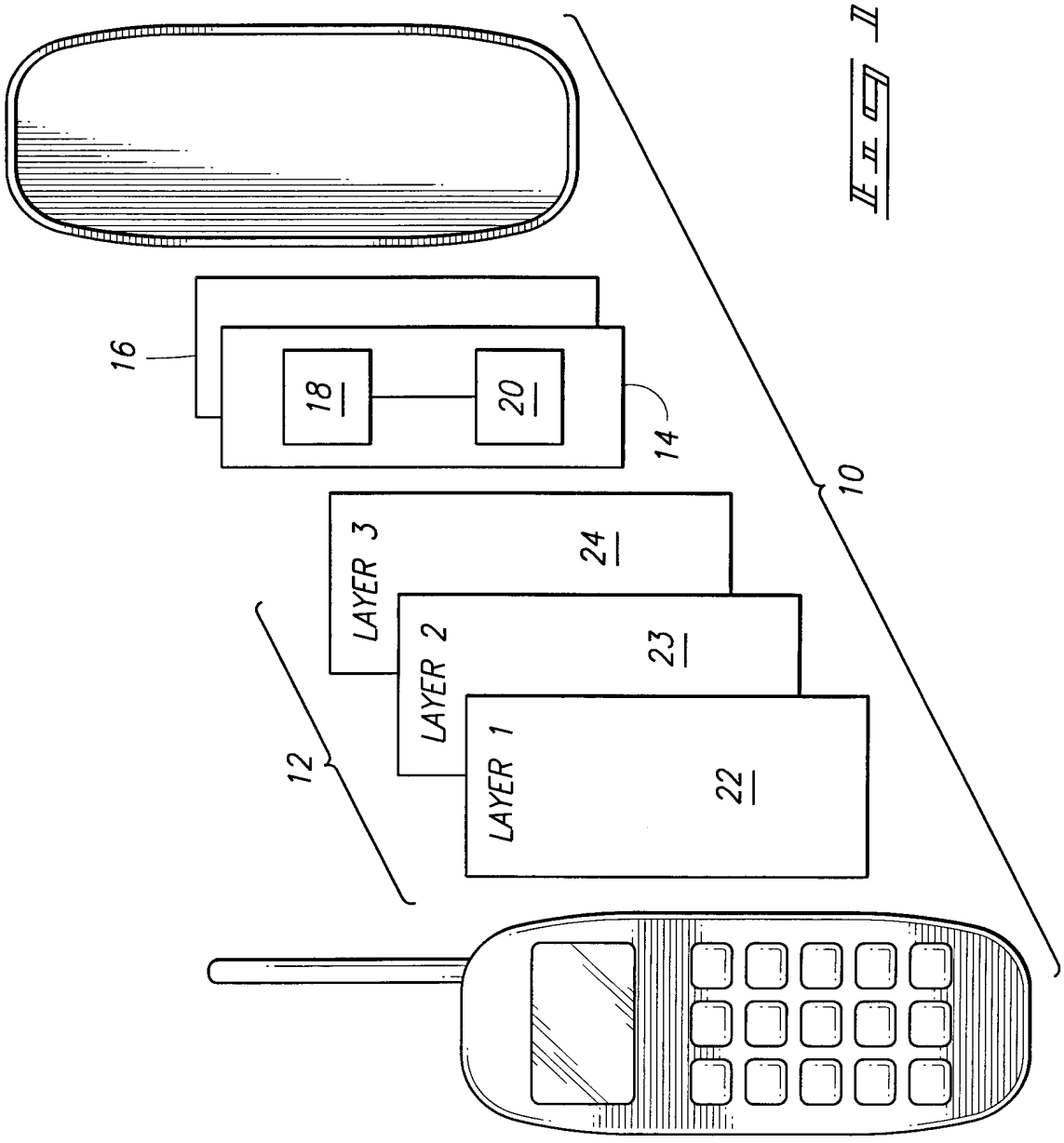


FIG. 1

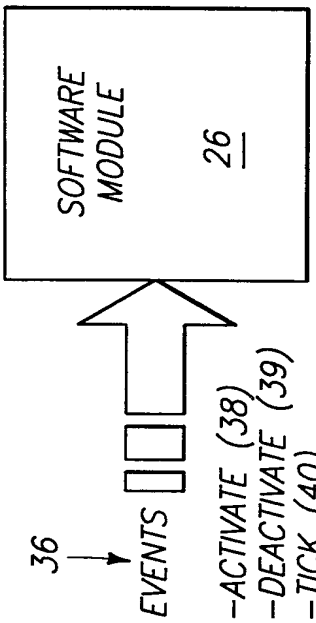

SOFTWARE MODULE
26
36
EVENTS
-ACTIVATE (38)
-DEACTIVATE (39)
-TICK (40)
-START (41)
-STOP (42)

Fig. 3

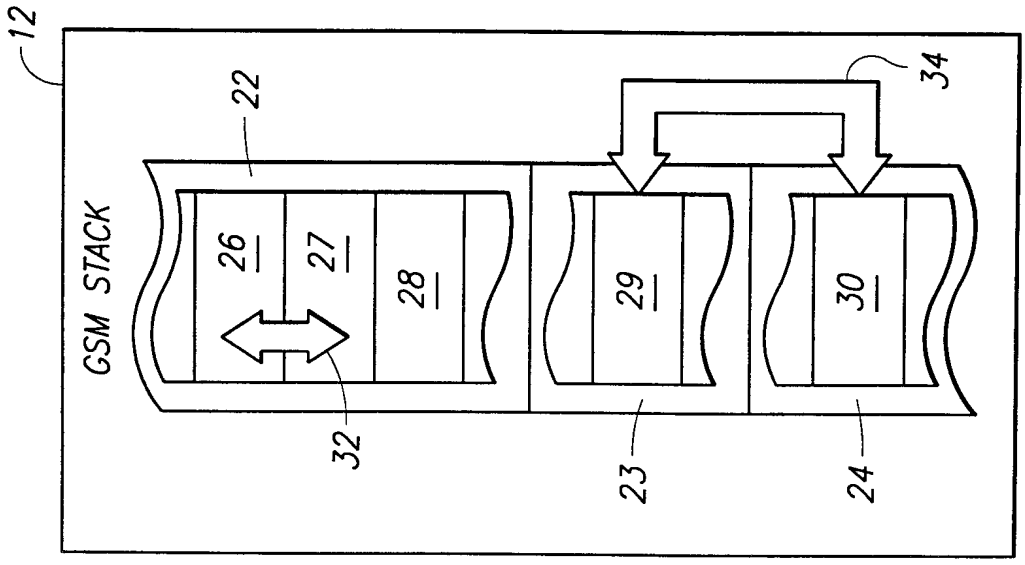

12
GSM STACK
22
26
27
28
29
30
32
23
24
34

Fig. 2

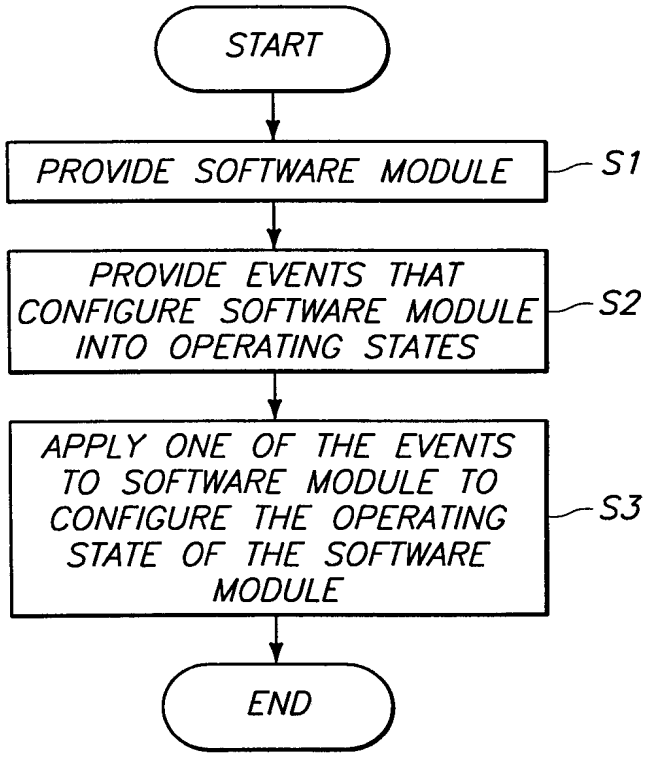

START
PROVIDE SOFTWARE MODULE
S1
PROVIDE EVENTS THAT CONFIGURE SOFTWARE MODULE INTO OPERATING STATES
S2
APPLY ONE OF THE EVENTS TO SOFTWARE MODULE TO CONFIGURE THE OPERATING STATE OF THE SOFTWARE MODULE
S3
END

Fig 4